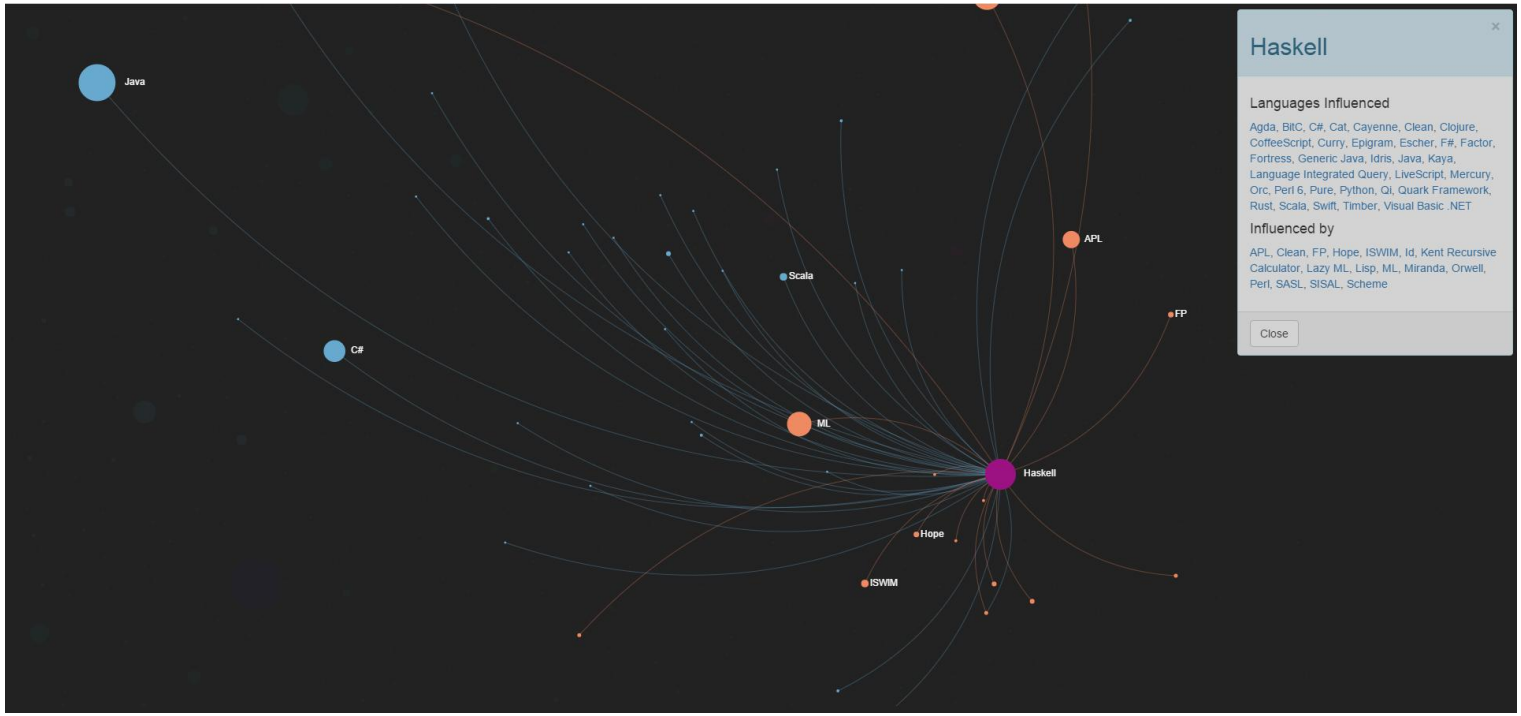
A Linguagem Haskell

Aluno: Rafael Teixeira de Oliveira
Matrícula: 2014.2.05028.11
Disciplina: Estruturas de Linguagens

Origens da Linguagem

- A linguagem Haskell surgiu em 1990, mas a história das linguagens funcionais (ainda a ser visto) começou bem antes disso.
  - Década de 1930: Church cria o Cálculo Lambda
  - Década de 1950: primeira linguagem com uso de cálculo Lambda, o LISP.
  - Década de 1960: primeira linguagem 100% funcional, o ISWIN.
  - 1978: a grande "alavanca" da programação funcional
  - 1987: conferência para criar linguagem funcional padrão
  - 1º/4/1990: lançamento da primeira versão de Haskell. O nome é em homenagem ao lógico Haskell Curry.

Influências do Haskell



Fonte: https://exploringdata.github.io/vis/programming-languages-influence-network-2014/#Haskell

Classificação do Haskell

- **Quanto ao paradigma:** é uma linguagem de programação do Paradigma Funcional, ou seja, uma linguagem de Programação Funcional.
  - O que é Programação Funcional? Paradigma onde a computação é feita com base em funções matemáticas
- **Quanto a tipagem:** é uma linguagem estática, mas sem precisar declarar tipo.
- **Compilada ou Interpretada?** Tem um interpretador interativo, mas ela também pode ser compilada.
- **Quanto ao uso:** mesmo com a proximidade com a matemática, ela é uma linguagem de propósito geral.

A Lazy Evaluation

- Também conhecida como “avaliação preguiçosa”.
- Conceito muito importante da programação funcional.
- Oposto da “avaliação ansiosa”.
- Computar (avaliar) expressão somente quando necessário.
- Importância dela: será vista posteriormente.

Avaliação da linguagem, quanto a redigibilidade e legibilidade

Haskell

```
module Main where

somaidades :: [Integer] -> Integer

somaidades lista=(sum lista)*5

main = print (somaidades
    [21,52,52,34,40,15,60,3])
```

C

```
#include <stdio.h>
int somaidades(int[]);
int somaidades(int idades[]){
    int soma=0;
    while(1){
            if((*idades)==NULL)
                    break;
            soma+=*idades;
            idades++;
    }
    return soma*5;
}

int main(){
    int
    idades[]={21,52,34,40,15,60,3,NULL};
    printf("%d",somaidades(idades));
}
```

4 linhas x 16 linhas

A expressividade da linguagem Haskell

- Como já foi visto anteriormente, o Haskell faz a avaliação das expressões de modo preguiçoso.
- Vantagem de evitar desperdício de recursos.
- Outra vantagem poderosa desta avaliação: o uso de listas infinitas.

- ghci> let a = [1..]
  ghci> let b = 5^2
  ghci> print b
  25

- Neste código, o 'a' não será avaliado, pois não está sendo usado, ficando apenas na definição.

- ghci> take 15(iterate (2*) 1)
  [1,2,4,8,16,32,64,128,256,512,1024, 2048,4096,8192,16384]

- Neste segundo código no interpretador, o ghci só avaliou a demanda requerida da lista infinita.

A expressividade da linguagem Haskell

- No segundo exemplo: o comando mais interno, *iterate*, passeará a lista infinita fazendo a i-ésima função composta definida no primeiro parâmetro, na i-ésima posição, com o valor, determinado no segundo parâmetro isto é, a lista ficaria assim, com a definição usada no exemplo:
  [1,f(1),f(f(1)),f(f(f(1))), ...].
- Contudo, o Haskell não processará toda a lista, avaliará somente a demanda desejada pela função *take*, que tem como finalidade recuperar os n primeiros da lista dada como parâmetro; isto é, os 15 primeiros.
- Tal recurso não é possível de ser usado em linguagens mais familiarizadas de todos

Conclusão

- A linguagem Haskell veio com o objetivo de "unificar' todas as linguagens funcionais que existiam até suas criação.
- Por sua boa redigibilidade, seus códigos são, quase sempre, condensados, pequenos, mas que, muitas vezes, pode pecar em sua legibilidade.
- A Haskell mostra que um problema pode ser visto como uma sequência de execução de funções matemáticas simples, que, passando o resultado de uma função para a próxima, pode-se chegar ao resultado esperado.

Fim! ☺